PAT-NO: JP410248014A

DOCUMENT-IDENTIFIER: JP 10248014 A

TITLE: IMAGE PROCESSING METHOD

PUBN-DATE: September 14, 1998

INVENTOR-INFORMATION:

NAME	COUNTRY
TAYLOR, ROBERT B III	
WINTER, KIRT ALAN	
CHOU, ROBERT	
NAIK, SACHIN	

ASSIGNEE-INFORMATION:

NAME	COUNTRY
HEWLETT PACKARD CO	N/A

APPL-NO: JP09331379

APPL-DATE: December 2, 1997

INT-CL (IPC): H04N001/40 , G06F003/12

ABSTRACT:

PROBLEM TO BE SOLVED: To reduce an amount of image data passing through a system during image processing.

SOLUTION: A user application registers a document including an image to be printed to an image **server and receives** an identifier used to identify a document from the image server. Then the user application gives the **identifier to a printer** driver 300 together with a print parameter that describes a desired aspect of a document to be printed. The printer driver 300 uses the identifier to reference the document and sends an image processing request to the image server 10, the image server 10 on the request processes the image of the document according to the print parameter to generate a printer image. Then the printer driver 300 extracts the processed

printer image consisting of the required image data only to be downloaded to printer 22, 24 from the image server 10.

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－248014

(43)公開日　平成10年(1998)９月14日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｈ０４Ｎ　1/40		Ｈ０４Ｎ　1/40	Ｚ
Ｇ０６Ｆ　3/12		Ｇ０６Ｆ　3/12	Ａ

審査請求　未請求　請求項の数１　ＯＬ　（全 14 頁）

(21)出願番号　特願平9－331379

(22)出願日　平成９年(1997)12月２日

(31)優先権主張番号　７６７，１２６

(32)優先日　1996年12月９日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　590000400
ヒューレット・パッカード・カンパニー
アメリカ合衆国カリフォルニア州パロアルト　ハノーバー・ストリート　3000

(72)発明者　ロバート・ピー・テイラー・サード
アメリカ合衆国98682ワシントン州バンクーバー、エヌ・イー、158プレイス　3519

(72)発明者　カート・アラン・ウィンター
アメリカ合衆国92026カリフォルニア州エスコンディド、ギルモア・プレイス　1358

(74)代理人　弁理士　岡田　次生

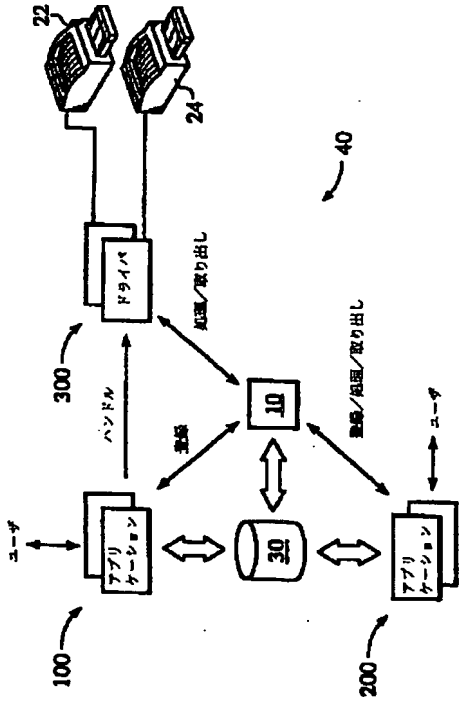
最終頁に続く

(54)【発明の名称】　画像処理方法

(57)【要約】

【課題】　画像処理の間にシステムを通過する画像データ量を減少させる。

【解決手段】ユーザ・アプリケーションは、本発明に従って備えられる画像サーバに印刷すべき画像を含むドキュメントを登録し、ドキュメントを識別する識別子を画像サーバから受け取る。その後、印刷されるドキュメントの所望の様相を記述する印刷パラメータとともに、識別子をプリンタ・ドライバに渡す。プリンタ・ドライバはドキュメントを参照する識別子を使用して、画像サーバに画像処理要求を送り、その要求に応じて、画像サーバが印刷パラメータに従ってドキュメントの画像を処理してプリンタ画像を生成する。次に、プリンタ・ドライバは、プリンタにダウンロードするために必要とされる画像データからのみ構成される処理済みプリンタ画像を画像サーバから取り出す。



1

【特許請求の範囲】
【請求項1】デジタル画像を取得するステップと、
上記デジタル画像の所望の様相を標示する変換パラメータを指定するステップと、
上記デジタル画像を画像サーバに登録するステップと、
上記登録ステップに対する応答として上記画像サーバから画像識別子ハンドルを受け取るステップと、
上記デジタル画像を識別する上記画像識別子と共に、上記変換パラメータに従って上記デジタル画像の画像データの上記所望の様相を生成するように画像処理要求を上記画像サーバに送るステップと、
生成された上記画像データを上記画像サーバから取り出すステップと、を含む画像処理方法。
【発明の詳細な説明】
【0001】
【発明の属する技術分野】本発明は、一般的にはドキュメント処理に関するもので、特に印刷のためドキュメントを準備する方法およびシステムに関するものである。
【0002】
【従来の技術】画像処理は、画像を捕捉するために使用されるハードウェアおよびソフトウェアの簡便さおよび普及によってごく一般的活動となった。インターネットの一般化およびアクセス可能性、および一層重要な点であるがワールド・ワイド・ウェッブ(WWW)の登場が画像の使用を顕著に増加させた。しかしながら、捕捉された画像およびその他のグラフィック・エレメントの処理は、テキスト・データの処理と比較して一般的に複雑である。例えば、カラー画像は、ビデオ・ディスプレイとプリンタの間のカラー合致や一定の特徴を抽出するためのエッジ強調などを必要とする。そのような操作はCPU集約的となる傾向があり、画像ファイル特にカラー画像に関連するデータは典型的には大量であるので、そのような操作はまた入出力集約的となりがちである。画像の使用の増加によって影響を受ける分野は、例えば、そのような画像を含むドキュメントの準備および印刷の分野である。
【0003】高度化された文書処理およびその他のデスクトップ・パブリッシング・プログラムの普及によって、ドキュメントは一般的にはもはや単にテキスト・データから構成されることはなく、むしろ種々の画像を含む。コンピュータ上での印刷のためそのようなドキュメントを準備するためには、拡大縮小、切り取り、鮮鋭化、種々の変換などを含むいくつかの画像処理操作を必要とする。それらの操作は、アプリケーション、または、プリンタに通常含まれるプリンタ・ドライバ・ソフトウェアによって実行される。画像をドキュメントに組み込む結果として、コンピュータ・システムは、グラフィック画像エレメントを表す大量のデータの処理のため高い度合いの負荷を経験する。ドキュメントの準備および印刷に対するアプローチはいくつかある。一部のアプ

2

リケーションは、印刷されるべきドキュメント画像の装置独立的ビットマップを準備する。ビットマップは次にその対応するプリンタ・ドライバ・ソフトを経由してプリンタに送られる。ほとんどの場合、アプリケーションはプリンタの必要条件に関する知識を持たない。例えば、ユーザはある種類の紙を使用する媒体品質画像を印刷することを望むかもしれない。これは、ドキュメント画像が一定の解像度およびビット深度をもって作成されることを必要とする。アプリケーションは、ユーザによって選択される印刷パラメータに応じて出力を提示するために必要な解像度やビット深度を判断することができない。そのような知識はプリンタ・ドライバに含まれている。従って、どのような種類のプリンタにも適応しなければならないアプリケーションは、保守的アプローチを取り、画像の高解像度ビットマップのコピーをプリンタ・ドライバへ送る。そこで、プリンタ・ドライバは、受け取った高解像度画像コピーに対して(例えば間引きのような)追加処理を実行して、当該プリンタにとって適切な解像度の出力を生成する。
【0004】この単純なアプローチは、非常に非効率な印刷方法である。アプリケーションは、プリンタおよび選択された印刷パラメータに関係なく、常にドキュメントの高解像度コピーを生成する。これは、不必要な大量のデータを変換し、ディスクの数回の入出力を伴う中間ファイルの作成を伴う過度のディスク活動を必要とする。そのような活動は、印刷ジョブが完了するまでCPUへのアクセスが止められるので、ユーザに悪い影響を与える。
【0005】別のアプローチにおいては、ドキュメント処理アプリケーションは、どのような画像処理能力がプリンタ・ドライバに存在するか判断するためプリンタ・ドライバに照会する。次に、アプリケーションはアプリケーションとプリンタ・ドライバの間で画像処理作業をどのように分担すべきか決定する。アプリケーションはドキュメントの適切な画像を作成しプリンタ・ドライバに画像のコピーを送る。プリンタ・ドライバは、受け取った画像のコピーに対して必要な追加処理を実行してドキュメントの最終的なプリンタ画像をプリンタへ出力する。出力解像度生成のようなプリンタ特有処理がプリンタ・ドライバによって実行されるので、プリンタ・ドライバ画像処理能力を知っていることは効率を改善する。アプリケーションは、もはや、すべてのプリンタに適応するため高解像度ビットマップを生成する必要がない。むしろ、アプリケーションは、切り取り、赤目補正、鮮鋭化などプリンタ・ドライバによって典型的にサポートされていない操作のような装置独立的操作のみを実行する。これによって、アプリケーションからドライバへ渡される画像ファイルは小さくなる。しかしながら、ユーザの観点からみれば、アプリケーションが画像を処理しプリンタ・ドライバに画像のコピーを渡し、その時点で

プリンタ・ドライバが受け取った画像のコピーに対するプリンタ特有の追加処理を実行してプリンタに生成したプリンター画像を出力するまで、ＣＰＵはなお占有される。

【０００６】待機を軽減するため、ほとんどのコンピュータは、スプーリングとして知られる技法を使用して印刷ジョブから可能な限り迅速にＣＰＵをユーザに返す。プリンタ画像、すなわちプリンタに実際に送られるビットマップ画像は、ディスクにコピーされ背景プロセスとしてプリンタに送られる。これによって、プリンタ・ドライバが、実際の出力が作成されるのを待たずにコンピュータの制御をユーザに返すことを可能にする。しかしながら、プリンタへの最終的プリンタ画像の伝送は、印刷ジョブの全体のわずかな部分を占めるにすぎず、ほとんどの努力と時間は、アプリケーションおよびプリンタ・ドライバが最初にドキュメントのプリンタ画像を生成するために費やされる。

【０００７】スプーリング技術を一歩進めたオペレーティング・システムがある。マイクロソフト・ウィンドウズ９５オペレーティング・システムの下で、改良メタファイル形式(すなわちenhanced metafile formatを略してＥＭＦと呼ばれる)スプーリングとして知られる方法が実施された。ＥＭＦスプーリングを使用すれば、オペレーティング・システム(以下単にＯＳと呼ぶ)は、プリンタ画像が現行のようにスプールされる場合と同じ方法で、アプリケーションがプリンタ・ドライバに対して行う呼び出しすべてをスプール・ファイルにスプールする。次に、ＯＳはスプールされたファイルを再生して、プリンタ・ドライバにスプールされた情報を送り出し、それによって印刷ジョブは完全に背景ジョブとして実行される。プリンタ・ドライバがその画像処理操作を実行するのをユーザはもはや待つ必要がないので、ＥＭＦスプーリングの下ではコンピュータの制御は一層迅速にユーザに返される。

【０００８】しかし、このようなＥＭＦスプーリングを使用しても、印刷の間システム性能に影響を与える要因がなお存在する。もっとも注意すべき要因は、大量のデータがシステムを通過するという事実である。8"x10"の写真の２４ビット・カラー・デジタル画像を例として考察すれば、そのような画像を３００dpiで表現するには約２０メガバイトのデータが必要とされ、６００dpiで表現するには約８０メガバイトが必要となる。ＥＭＦスプーリングの場合、ＯＳは、アプリケーションからプリンタ・ドライバに対して行われる呼び出しと共に、画像を表現する２０または８０メガバイトのデータのコピーをもスプールする。スプールされるファイルはディスクに書き込まれる。再生の間、(２０メガバイトの画像データを含む)スプールされたファイルがディスクから読み戻され、プリンタ・ドライバに渡される。プリンタ・ドライバがプリンタ画像を準備する際、一時的ファイルが作成されディスクに書き込まれ、ディスクから読み戻され、再びディスクへ書き込まれる等の動作が行われる。このようなディスク活動のすべては、ＯＳの応答時間および性能全体に影響を与える。ユーザは、非常に遅いかまたは反応しないシステムに出会うこととなる。

【０００９】

【発明が解決しようとする課題】以上の記述は、印刷ジョブの処理の間に経験される隘路に焦点をあてたが、同様の隘路は、画像処理に一般的に存在する。画像処理の間にＯＳを通過するデータの量を最小にする方法およびシステムが必要とされる。特に、印刷ジョブを一層効果的に処理する方法およびシステムが必要とされる。更に、印刷ジョブのような画像処理の間にＯＳがかかわるディスク活動を最小にすることもまた必要とされる。

【００１０】

【課題を解決するための手段】本発明に従って、所望の様相の画像を生成するために行われる画像処理は、画像サーバに画像を登録することから始まる。画像サーバは、画像サーバ内で画像を識別するハンドルを返す。次に、画像の所望の様相に従った画像処理要求が画像サーバに送られる。次に、処理された画像が画像サーバから取り出される。

【００１１】本発明の１面において、ドキュメントを印刷する方法およびシステムが提供される。この方法およびシステムは、ドキュメントを画像サーバに登録し、その後ドキュメントを識別する識別子を該画像サーバから受け取ることを含む。印刷されるべきドキュメントは、ドキュメントの位置を特定するファイル記述子を画像サーバに渡すことによって、画像サーバに登録される。ファイル記述子は、典型的にはディスクに記憶されるドキュメントのファイル名であるが、ドキュメントが記憶される共有メモリにおける位置を指し示すポインタでもよい。その後、印刷されるドキュメントの所望の様相を記述する印刷パラメータとともに、上記識別子がプリンタ・ドライバに渡される。ドキュメントを参照する識別子を使用して、プリンタ・ドライバは、画像サーバに画像処理要求を送り、その要求に応じて、画像サーバが印刷パラメータに従ってドキュメントの画像を処理することによってプリンタ画像を形成する。最後に、プリンタ・ドライバは、プリンタにダウンロードするために必要とされる画像データからのみ構成される処理済みプリンタ画像を画像サーバから取り出す。

【００１２】印刷の間システムを通過するデータ量は最小限に保たれる。例えば、ある１つのプロセスのアドレス空間から別のプロセスのアドレス空間にドキュメントを表現する大量のデータをコピーする代わりに、識別子が使用される。識別子はドキュメントを識別するのに役立ち、それがドキュメントそのものではなくドキュメントの識別子であるので、通過するデータ量は顕著に減少する。

【００１３】本発明の別の特徴は、最終的プリンタ画像を形成するために必要とされる画像部分だけが画像サーバによってアクセスされる点である。このようにして、ドキュメントに含まれる画像の取り扱いの量は最小限にとどめられる。その結果、プリンタ画像の形成の間にディスクとの間で行われるデータ保存および読み出しのためのディスク活動が削減される。加えて、ＣＰＵに対する負荷も減少する。これは、ユーザがプリント・コマンドを出すターンアラウンド・タイムを迅速化しシステム性能を向上させるという全般的効果を持つ。

【００１４】本発明を実施するシステムにおいては、印刷ジョブを始動するユーザ・アプリケーションは、画像サーバおよびプリンタ・ドライバの両方とは分離独立して稼働する。同様に、ＯＳがマルチタスクをサポートしている限り、画像サーバおよびプリンタ・ドライバは、別々のプロセスとして、あるいは並列プロセスとして、あるいは独立実行プロセスとして稼働することができる。画像サーバがドキュメントを登録するようにアプリケーション(すなわち文書処理アプリケーション)から要求を受け取ると、画像サーバは、他のプロセスがドキュメントにアクセスするのを防止するためドキュメントをロックする。これによって、画像サーバが所望のプリンタ画像を生成している間のドキュメントの保全が保証される。しかしながら、その後ロック解除要求を受け取り次第、画像サーバはドキュメントの一時的コピーを作成して、オリジナルのロックを解くことができる。ドキュメントは、画像サーバによって先ず準備される識別子という手段によって参照されているので、画像サーバの外側のプロセスはコピーによって影響を受けない。

【００１５】本発明の別の局面において、複数のアプリケーションがロードされ、それぞれのアプリケーションが印刷されるべきドキュメントを選択し、印刷ジョブを始動することができる。同様に、複数のプリンタ・ドライバをロードして、それらを独立したプログラム単位として実行させることも可能である。このようにして、多数の種類のプリンタをサポートすることができる。そのようなプリンタ・ドライバの各々は、その対応するプリンタと通信する能力を持つようにプログラムされるか、さもなければ、そのような構成に配置される。

【００１６】本発明の更に別の局面において、画像サーバおよびプリンタ・ドライバは、複数のネットワーク上に存在するように構成できる。アプリケーションは画像サーバまたはプリンタ・ドライバと同じコンピュータに配置されることも、あるいは別のコンピュータに配置されることもできる。これによって、１つのファイル・サーバおよび複数のプリンタに関連して構成される複数のワークステーションを有するサイトを構成する場合、最大限の柔軟性が与えられ、同時に本発明によって可能とされる性能上の利点が実現する。

【００１７】本発明は、発明の課題を解決する手段として、デジタル画像を取得するステップ、上記デジタル画像の所望の様相を標示する変換パラメータを指定するステップ、上記デジタル画像を画像サーバに登録するステップ、上記登録ステップに対する応答として上記画像サーバから画像識別子ハンドルを受け取るステップ、上記デジタル画像を識別する上記画像識別子と共に、上記変換パラメータに従って上記デジタル画像の画像データの上記所望の様相を生成するように画像処理要求を上記画像サーバに送るステップ、および、生成された上記画像データを上記画像サーバから取り出すステップからなる画像処理方法を含む。

【００１８】

【発明の実施の形態】本発明を実施するために考案された最良の形態が、図１のシステム・ブロック図に示されている。コンピュータ・システム４０において、１つまたは複数のユーザ・アプリケーション１００、２００が、ユーザ・インターフェースを介してドキュメントを生成し処理する。そのようなアプリケーションには、文書処理プログラム、作図プログラム、表計算プログラム、ドキュメント構成プログラム、写真画像プログラムなどが含まれる。本明細書で使用する用語「ユーザ」は、アプリケーションと対話する人間か、または例えば到来ファックスを自動的に受け取るファックス機の機能を果たすソフトウェア、画像走査ソフトウェア、自動データ検索ソフトウェアなどのアプリケーションと対話する自動エージェントのいずれかとして定義される。アプリケーション１００は、必要の都度データを記憶し取り戻すためデータ記憶装置３０へアクセスする。更に、データ記憶装置３０には、アプリケーションのランダム・アクセス・メモリ(ＲＡＭ)のアドレス空間内の区域が含まれ、それら共有メモリ内の異なる区域が参照される場合もある。画像処理サーバ１０は、画像内包ドキュメントを操作する画像処理能力を提供する。アプリケーション１００、２００は、画像サーバ・インターフェースを経由して画像サーバ１０と通信する(詳細は後述)。アプリケーションと同様に、画像サーバ１０はデータ記憶装置３０へのアクセスを行う。

【００１９】図１において、いくつかのアプリケーション１００が、印刷装置２２、２４に接続するプリンタ・ドライバ３００と通信している。プリンタ・ドライバ３００の各々は、特定のタイプのプリンタ装置に関して特別にプログラムされる。例えば、プリンタ装置２２がプロッタであり、プリンタ装置２４がレーザ・プリンタであるかもしれない。あるいはまた、プリンタ装置２２、２４の両方がインクジェット式プリンタであるが製造業者は異なり、プリンタ各々と通信できるドライバを必要とするかもしれない。

【００２０】本発明の１つの局面において、複数のプリンタ・ドライバ３００が、独立して実行するプログラム単位としてロードされる。すなわちそれらプリンタ・ド

ライバの各々は独立プロセスである。プロセスの間の通信は、RPC OLEおよびCOMのような周知のプロセス間通信(すなわちinterprocess communicationでその頭文字をとってＩＰＣと呼ばれる)技術を介して提供される。ほとんどのオペレーティング・システム(ＯＳ)は少くとも何らかの形式のマルチタスキングをサポートし、プロセス間のＩＰＣを提供する。代表的ＯＳは、ＵＮＩＸ ＯＳの多数の変形、マッキントッシュＯＳ、ウィンドウズ３.Ｘ、ウィンドウズ９５およびマイクロソフトＮＴを含む。当然のことながら、このリストは単に典型的例を示すものであり、それらに制限する意図はない。本発明は、ＩＰＣをサポートするその他のマルチタスキング・システム上で効果的に実施することができる。

【００２１】本発明に従って画像およびドキュメントを印刷する場合の典型的実行ステップが図２の流れ図に示されている。図２で示されるシーケンスは、図１で示されるアプリケーション１００およびプリンタ・ドライバ３００に対応する。最初に、ステップ６０において、データ記憶装置３０またはアプリケーション・ユーザによって与えられる入力からアプリケーション１００がファイルに読み取ることによって、ドキュメントが取得される。ドキュメントは、ディスク上のファイル名またはドキュメントが記憶されるメモリのアドレス・ポインタによって、識別される。ドキュメントの取得に加えて、アプリケーション１００は、印刷されるドキュメントの所望の様相を記述する印刷パラメータをユーザから取得する。

【００２２】次に、ステップ６２において、アプリケーション１００は、ドキュメントのファイル名またはアドレス・ポインタを画像サーバに送ることによって、ドキュメントを画像サーバ１０に登録する。上述のように、アプリケーション１００、２００および画像サーバ１０は独立して実行するプロセスとすることもできる。従って、各プロセスは、一般的に他のプロセスによって直接アクセスされないそれ自身の論理アドレス空間を持つ場合がある。このように、アプリケーションがアドレス・ポインタを介してドキュメントを登録する場合、ドキュメントは少なくとも２つのプロセスによってアクセスされることができる共有メモリ区域に置かれなければならない。同様に、データは、動的連結ライブラリ(すなわちＤＬＬ)を使用する場合のようにデータが１つのアドレス空間に存在する時、アドレス・ポインタが使用される。

【００２３】ドキュメント識別子を受け取り次第、画像サーバ１０は、アプリケーション１００によって提供された識別子とは別に、ドキュメントを識別するハンドルを作成する(ステップ６４)。用語「ハンドル」は典型的には特定のデータ構造を暗示するが、本明細書における「ハンドル」という用語は、そのようなデータ構造に限定するように意図されてはいない。むしろ、本明細書で与えられる記述と整合する「ハンドル」のいかなる実施も考慮されている。プロセスはステップ６６へと進み、画像サーバは、他のアプリケーションによるアクセスを防止するためドキュメントをロックする。ドキュメント・ハンドルおよびドキュメント・ロック機構によって、ドキュメントが画像サーバによって処理されている間のドキュメントの保全および整合性が保証される。例えば、第２のアプリケーションがドキュメントを修正しようとすると、その第２のアプリケーションは、先ず、画像サーバにアンロック要求を送らなければならない。画像サーバはその要求に応答して、ドキュメントから局所的コピーを作成しオリジナルのロックを解く。その後画像サーバはその局所的コピーを参照し、第２のアプリケーションがアクセスできるようにオリジナルのドキュメントを解放する。

【００２４】続いて、アプリケーション１００は、画像サーバによって作成されたハンドルを受け取り(ステップ６８)、ハンドルおよび指定された印刷パラメータをプリンタ・ドライバ３００へ送る(ステップ９０)。次に、プリンタ・ドライバ３００は、印刷パラメータに従ってハンドルによって指定された画像を処理するように画像サーバ１０へ１つまたは複数の画像処理要求を送る(ステップ７０)。次に、プリンタ・ドライバは、処理された画像データを画像サーバから取り出し(ステップ７２)、対応するプリンタ２２、２４へ取り出した画像データを送る(ステップ９２)。最後に、ドキュメントのロックが解かれる(ステップ７４)。

【００２５】画像サーバが上述のようにドキュメントの局所的コピーを作成するとすれば、画像処理要求は局所的コピーに向けられる。プリンタ・ドライバがドキュメントを参照するためドキュメント・ハンドルを使用しているので、コピーするという事実はプリンタ・ドライバにとって完全に透過的である。印刷ジョブの完了次第、画像サーバは局所的コピーを削除する。

【００２６】一般的に、画像サーバ１０は、本発明において画像処理操作の大部分を提供する。しかしながら、画像サーバの処理能力をいかなるタイプのプリンタにも従属しないような一般的処理に制限することが望ましい場合もある。プリンタ特有の画像要件は、プリンタに送られる最終的プリンタ画像を作成するためステップ７２において画像データを取り出し次第、プリンタ・ドライバによって実行することができる。

【００２７】図２は、本発明をサポートしないシステムにおける互換性問題に対処する付加的ステップ８０乃至８４を含む。画像サーバ１０へのドキュメント登録に先立ち、アプリケーション１００は、先ず、プリンタ・ドライバが画像サーバに対するインターフェースを有しているか否かを判断するためプリンタ・ドライバに照会を送る。回答が否定的であれば、アプリケーションは従来技術の印刷方法に従って処理を進める。これは、典型的

には、アプリケーションが印刷されるべきドキュメントの装置独立ビットマップ(すなわちＤＩＢ)画像を生成し(ステップ８２)、引き続き行われる印刷のためそのＤＩＢ画像をプリンタ・ドライバへ送る(ステップ８４)ことを意味する。しかしながら、プリンタ・ドライバが画像サーバ・インターフェースをサポートしていれば、アプリケーションは、上述のようにステップ６２乃至９２を通して処理を進める。

【００２８】別の実施形態においては、本発明の方法は、図２において６０から７２までの記号によって示されるステップだけを含む。このシーケンスは、図１のアプリケーション２００に対応する。この代替実施形態は、本発明の一般化であり、上述の印刷処理はその特別ケースである。一般化された方法においては、アプリケーション２００自体が画像サーバと通信して、ドキュメントに関する所望の画像処理演算処理を実行させる(ステップ７０)。加えて、結果として生ずる処理された画像データは必ずしも印刷されるというわけではなく、単にデータ記憶装置３０に記憶するだけの場合もあり、またビデオ端末上に表示されることもある。

【００２９】本発明を構成するコンポーネントを以下説明する。多数のソフトウェア・モジュールが、本システムの主要コンポーネントを構成する。従って、本発明を実施するため、ＣおよびC++を含む多くの既存のプログラミング言語を使用することができる。ソフトウェア・モジュールの関数は、各モジュールに対するアプリケーション・プログラマ・インターフェース(すなわちＡＰＩ)として定義され、特定の実施詳細は、ＡＰＩを実施するために必要とされる演算を実行するコードを単に作成することにすぎない。しかしながら、場合によっては、実際のＯＳという観点から演算を説明することが必要かもしれない。そのような場合に必要とされる文脈を準備するためにはウィンドウズ９５オペレーティング・システム環境が使用される。

【００３０】図３に示されるように、アプリケーション１００は、典型的には、アプリケーション特有コード１０１、プリンタ・ドライバ３００と通信するプリンタ・ユーティリティ３０２、記憶装置３０への読み書きを行うＩ／Ｏユーティリティ３２、および画像サーバ１０と通信する画像サーバ・ユーティリティ１２を含む。アプリケーション特有コード１０１は、当然、アプリケーションに従って変わる。上述の通り、アプリケーションには、文書処理、図面プログラム、表計算、デスクトップ・パブリッシング・プログラムなどの種々のタイプが含まれる。

【００３１】プリンタ・ドライバ３００は、印刷装置と通信するドライバ特有コード３０１を含み、更に画像サーバと通信する画像サーバ・ユーティリティを含む。印刷装置に対する通信インターフェースを備えることに加えて、ドライバ特有コード３０１は、プリンタに特有の画像処理演算を実行する場合もある。従って、画像サーバ１０は、装置に依存しない画像処理演算だけを行えばよい。

【００３２】画像サーバ１０は、Ｉ／Ｏユーティリティ３２に加えて、解像度変更、画像拡大縮小、切り取り、赤目補正、回転、特徴抽出、輪郭強化などの画像処理操作を実施する画像サーバ固有コード１１を含む。そのような操作は、コンピュータ画像処理およびコンピュータ・グラフィックス技術分野において周知のものである。画像サーバは、ＪＰＥＧ、ＭＰＥＧ、ＧＩＦ、ＴＩＦＦ等々を含む多くのタイプの画像形式をサポートすることができるように設計される。どのようなタイプの形式が画像サーバによってサポートされるかは、本発明にとって重要ではなく、むしろ、形式の選択は、画像サーバに要求される能力、コードの複雑さ、画像形式の複雑さ、処理能力、記憶容量、操作環境、システム・コストなどという要因の重さに依存する。

【００３３】Ｉ／Ｏユーティリティ３２は、ファイルのオープン／クローズおよびファイルの読み書きを含む標準的動作を実行する。そのような動作はソフトウェア業界において十分理解されているものであるので、これ以上の説明は必要ではない。

【００３４】プリンタ・ユーティリティ３０２は、アプリケーション１００がプリンタ・ドライバ３００と通信して印刷ジョブを始動することを可能にする。プリンタ・ユーティリティには、プリンタ・ドライバ３００が画像サーバ・インターフェースをサポートしているか否かをアプリケーション１００が判断する(図２のステップ８０参照)ことを可能にする機能が含まれる。ウィンドウズ９５の下では、これは、例えばIS_SUPPORTコード(画像サーバ・サポート)に関するExtEscape()システム呼び出しを行うことによって達成できる。画像サーバ１０をサポートするプリンタ・ドライバ３００は、IS_SUPPORTコードを認識し、肯定的応答を行う。従って、ExtEscape(QUERYESCSUPPORT, IS_SUPPORT)に対するシステム呼び出しの戻り値は正である。そうでない場合は、画像サーバがサポートされていないことを示す値であるゼロが戻される。

【００３５】画像サーバがサポートされていなければ、アプリケーションは、従来技術における現行の方法で処理を続ける。すなわち、アプリケーションはＤＩＢ画像を生成してプリンタ・ドライバ３００にその画像を送る。これはウィンドウズ９５の下で、StretchDIBits()システム呼び出しを行ってＤＩＢ画像へのポインタを渡すことによって実行される。一方、プリンタ・ドライバが画像サーバをサポートしていれば、アプリケーションは、画像に対するハンドルを取得し(図２のステップ６８参照)、StretchDIBits()システム呼び出しを行ってＤＩＢ画像の代わりにそのハンドルを渡す。

【００３６】画像サーバ・ユーティリティ１２は画像サ

ーバに対するインターフェースを定義する。このように、画像処理操作は、ユーティリティに対して適切な呼び出しを行うことによって達成される。画像サーバ・ユーティリティに関するアプリケーション・プログラマ・インターフェース(ＡＰＩ)は後述される。これらのＡＰＩは、画像サーバの機能性を定義し、使用されるＯＳおよびプログラミング言語に依存する画像サーバを実施する原始コードのサンプルを提供する必要性を取り除く。当業者がＡＰＩによって定義される関数を実施するコードを生成することができる点は明白である。

【００３７】上述のように、画像サーバ１０は種々の画像形式をサポートすることができる。画像サーバが画像サーバ・ユーティリティ１２を通してアクセスされるので、新しい形式が定義される都度、画像サーバは、追加画像形式をサポートするように容易に拡張することができる。画像サーバ・ユーティリティ１２の存在は、画像サーバを使用するアプリケーション／ドライバに対してこのような拡張を完全に透過的にするので、画像サーバが新しい画像形式をサポートするように更新される時、後方互換性が維持される。

【００３８】図３には、画像サーバ・ユーティリティに対するＡＰＩ定義が示されている。図３は、ユーティリティが、アプリケーション１００とプリンタ・ドライバ３００の間に分けられていることを示している。しかし、これは必ずしも必要であるというわけではない。特定の用途に従って、リストされたユーティリティのすべてが１つのアプリケーション内に配置されることも、多数のアプリケーションの間に分けられることもある。以下にＡＰＩをアルファベット順に記述する。

【００３９】CloseImage

機能：画像ファイルの処理が完了したことを画像サーバに通知する。画像サーバは、画像ファイルのロックを解き、他のプロセスによるそのファイルへのアクセスを可能にする。

パラメータ：

・ihImageHandle－画像ファイルを識別する。

戻り値：

・TRUE(真)－関数が成功した。

・ISE_NOTREGISTERED－ihImageHandleが現在登録されている画像ファイルに対応しない。

【００４０】ConfigureImage

機能：(形式オプションの)画像処理演算が画像に対して実行されるべきことを画像サーバに通知する。

パラメータ：

・ihImageHandle－処理されるべき画像ファイルを識別する。

・ipImageProcOptions－所望の画像処理オプションを含むデータ構造。

戻り値：

・TRUE－関数の成功。

・ISE_FORMAT_NOTSUPPORTED－指定された形式は画像サーバによってサポートされていない。

・ISE_NOTREGISTERED－ihImageHandleが現在登録されている画像ファイルに対応しない。

【００４１】OpenImage

機能：画像処理演算が(ConfigureImage()によって)指定された画像に関して実行されるべきことを画像サーバへ通知する。

パラメータ：

・ihImageHandle－処理されるべき画像ファイルを識別する。

戻り値：

・TRUE－関数の成功。画像サーバは、ihImageHandleに対する画像処理要求を受ける準備ができている。

・ISE_NOTREGISTERED－ihImageHandleが現在登録されている画像ファイルに対応しない。

【００４２】QueryFormatSupport

機能：指定された画像形式を画像サーバがサポートしているか否かを画像サーバに照会する。指定された画像形式に関して画像サーバによってサポートされる追加処理能力を示す情報が提供される。

パラメータ：

・dwImageFormat－当該画像形式。例えばＴＩＦＦ、ＪＰＥＧ、ＭＰＥＧ、ＧＩＦ。

・lpFormatData－画像サーバによって提供される追加処理能力(例えば合成)を示すデータ構造へのポインタ。

戻り値：

・ISE_FORMAT_SUPPORTED－指定された形式が画像サーバによってサポートされている。lpFormatDataがヌルのポインタでなければ、追加形式機能が画像サーバによって指定される。

・ISE_FORMAT_NOTSUPPORTED－指定された形式は画像サーバによってサポートされていない。

【００４３】QueryImage

機能：指定された画像ファイルに関する情報を提供する。

パラメータ：ihImageHandle－情報が必要とされる画像ファイルを識別する。

lpImageInfo－画像情報データ構造へのポインタ。画像サーバによって記入される。

戻り値：

・TRUE－関数の成功。

・ISE_NOTREGISTERED－ihImageHandleが現在登録されている画像ファイルに対応しない。

【００４４】RegisterImage

機能：指定された画像ファイルをロックしコピーして、ロック／コピーした画像ファイルに対するプライベート・ハンドルを戻す。ハンドルはファイルに対する以後のアクセスにおいて使用される。

パラメータ：

・lpszImageFileName－画像ファイルのパス／ファイル名を指す。別の方法としては、バッファへのポインタによって画像ファイルが渡される。この場合lpszImageFileNameはヌルのポインタである。
・lpJobData－印刷装置の名前のような適切なジョブ・データを含む構造をポイントする。
・dwImageFormat－指定された画像ファイルの画像形式を指定する。
・lpImageHandle－(画像サーバによって生成される)画像ハンドルがコピーされるバッファをポイントする。
・lpImageData－画像ファイルを含むバッファをポイントする。別の方法としては、パス／ファイル名によって画像ファイルは参照される。この場合lpImageDataはヌルのポインタである。
・lpImageDataHeader－lpImageDataによってポイントされるデータの形式を定義するデータ構造をポイントする。lpImageDataがヌル・ポインタであればこれはヌル・ポインタである。
・dwOptions－画像ファイルが画像サーバに登録される方法を指定する。有効な値は以下を含む。IS_TEMPFILE：lpszImageFileNameによって指定されるファイルは、CloseImageへの呼び出しが行われる時画像サーバによって削除されるべき一時的ファイルである。IS_COPYNOW：lpszImageFileNameによって指定されるファイルは一時的ファイルではなく、画像サーバは、ファイルをロックするのではなく画像ファイルからコピーを作成しなければならない。
戻り値：
・TRUE－関数の成功。lpImageHandleは、画像に関する(画像サーバによって作成された)有効なハンドルをポイントする。
・ISE_OUTOFSPACE－要求された画像ファイル・コピーを作成できない。lpImageHandleの内容は不確定。
・ISE_FORMNOTSUPP－dwImageFormatで指定された形式はこの画像サーバ・バージョンではサポートされていない。
【００４５】RetrieveImageBand
機能：ConfigureImage()に対する先行呼び出しの結果として画像サーバによって生成された処理画像の画像データを要求する。
パラメータ：
・ihImageHandle－画像ファイルを識別する。
・XSrc－画像サーバによって生成された画像データの所望の部分のX座標(ピクセル単位)。
・YSrc－画像サーバによって生成された画像データの所望の部分のY座標(ピクセル単位)。
・cxSrc－画像サーバによって生成された画像データの所望の部分の幅(ピクセル単位)。
・cySrc－画像サーバによって生成された画像データの所望の部分の高さ(ピクセル単位)。

・lpBandBuffer－画像サーバが生成された画像データの要求部分を書き込むべきバッファを指すポインタ。
・lpBandHeader－lpBandBufferに記憶されるべき画像データの形式を指定するデータ構造を指すポインタ。
戻り値：
・TRUE－関数は成功。
・ISE_PROCERROR－画像サーバが処理エラーに遭遇。
・ISE_NOTCONFIGURED－処理演算は実行されなかった。
・ISE_NOTREGISTERED－ihImageHandleが現在登録されている画像ファイルに対応しない。
【００４６】UnlockImage
機能：lpszImageFileNameに関するファイルのロックを、現在画像サーバによってロックされていれば、解除する。この場合、画像サーバは画像ファイルから一時的コピーを作成する。
パラメータ：
・lpszImageFileName－ロックを解除すべき画像ファイルのファイル名。
戻り値：
・TRUE－関数の成功。lpszImageFileNameはロック解除された。
・ISE_OUTOFSPACE－lpszImageFileNameのロックを解除するため画像ファイルのコピーを作成することができなかった。画像ファイルは画像サーバによってロックされたままである。
・ISE_NOTUNLOCK－lpszImageFileNameのロックを解除することができなかった。これは、ファイルのロックが画像サーバによって解除されなかったことを典型的に示す。
【００４７】以下の疑似コード(表１)は、上記の呼び出しが典型的ドキュメント処理アプリケーションにおいてどのように出現するかを示す。
【００４８】
【表１】

```
:
:
/* ＪＰＥＧがサポートされているか？ */
dwImageFormat <— JPEG画像形式
QueryFormatSupport( dwImageFormat, lpFormatData )
if( 戻り値 = = ISE_FORMAT_NOTSUPPORTED ) exit
/* 画像ファイル登録 */
lpszImageFileName <-- ドキュメント・ファイル名
lpJobData <-- ユーザ指定プリンタ、プリンタ・オプション
dwImageFormat <— JPEG画像形式
lpImageData <-- ヌル
lpImageDataHeader <-- ヌル
dwOptions <-- IS_COPYNOW /* ファイルをロックしない */
RegisterImage( lpszImageFileName,lpJobdata,dwImage
```

Format,lpImageHandle,lpImageData,lpImageDataHeader,dwOptions)
if(戻り値 ! = 真) exit
/* 印刷パラメータとともにプリンタ・ドライバにハンドル(lpImageHandle)を送る */
:
:

【００４９】以下の擬似コード(表２)は、画像サーバ・ユーティリティがプリンタ・ドライバ・ソフトにどのように出現するかを示す。

【００５０】

【表２】

```
:
:
/* 画像処理の開始を画像サーバに伝える */
ihImageHandle <-- ユーザ・アプリケーションから受け取った画像ハンドルOpenImage( ihImageHandle )
if ( 戻り値 = = ISE_NOTREGISTERED )ユーザ・アプリケーションにハンドル無効を通知する
exit
/* 画像処理を実行する */
すべての印刷パラメータ毎に以下を実行
ipImageProcOptions <-- 印刷パラメータ
ConfigureImage( inImageHandle,ipImageProcOptions )
if( 戻り値 = = ISE_FORMAT_NOTSUPPORTED )ユーザ・アプリケーションに印刷パラメータ無効を通知する
exit
続行
/* 処理された画像の画像データを取得してプリンタに送る */
XSrc <-- 処理画像の初期セグメントのＸ座標
YSrc <-- 処理画像の初期セグメントのＹ座標
cxSrc <-- 画像部分の幅
cySrc <-- 画像部分の高さ
lpBandHeader <-- GIF形式の処理済み画像を取り出す
処理画像データのすべてのセグメントに関して以下を実行
RetrieveImageBand( ihImageHandle,Xsrc,Ysrc,cxSrc,cySrc,lpBandBuffer,lpBandHeader )
lpBandBufferによってポイントされるセグメントをプリンタへ送る
処理画像の次のセグメントへ増分
Xsrc <-- 次のセグメントのＸ座標
Ysrc <-- 次のセグメントのＹ座標
続行
:
:
CloseImage( ihImageHandle )
:
:
```

【００５１】本発明の１つの実施形態が図４に示されている。図１に示されている本発明のコンポーネントが１つのコンピュータに配置されているのに対して、図４では図１と同じコーンポーネントがネットワーク化された複数のコンピュータに分散されている。コンピュータ４２乃至４８は１つのネットワーク５０によって相互接続されている。ネットワーク５０は、ＬＡＮ(ローカル・エリア・ネットワーク)におけるイーサネット・バックボーンであっても、あるいはＷＡＮ(ワイド・エリア・ネットワーク)における電話ネットワークであってもよい。より一般的には、コンピュータ４２乃至４８は、ＬＡＮおよびＷＡＮのいくつかの組合せによって相互接続されることができる。本発明は、どのようなネットワーク構成ででも実施することができ、特定の１つのアーキテクチャに限定されない。

【００５２】各コンピュータは、ネットワーク・インターフェース５２を経由してネットワーク上で通信する。インターフェース５２は、特定のコンピュータがネットワーク上の別のコンピュータにアクセスすることを可能にするために必要とされるハードウェアおよびサーバ・ソフトウェアを含む。上述のように、アプリケーション１００、画像サーバ１０およびプリンタ・ドライバ３００は別々のプロセスであり、それらプロセスの間の通信は適切なＩＰＣコールによって実行されるという点に注意する必要がある。ネットワーク構成においては、これらプロセスがネットワークにわたって動作することができるようにするため、ＩＰＣ呼び出しは、同等の遠隔手続き呼び出し(すなわちＲＰＣ)に置き換えられなければならない。ＲＰＣの実施は一般には特定のＯＳ能力に依存するが、それはプログラミング分野の通常の技術を有する者の知識範囲に入るものである。

【００５３】図４は、ディスク・サーバの機能を果たすコンピュータ４４を示す。このディスク・サーバに本発明の画像サーバ１０が含まれる。また、プリンタ２２、２４をドライブするプリンタ・ドライバ３００'がディスク・サーバ４４にロードされる。別の１つのコンピュータ４８は、主にプリンタ２６、２８のためのプリンタ・サーバの役目を果たす。コンピュータ４２上のユーザ・アプリケーション１００は、ディスク・サーバ４４上で動いているサーバ・ソフトウェアを経由してディスク３０上のファイルにアクセスすることができる。コンピュータ４６上で動いているユーザ・アプリケーション２００は同様の形態で動作する。

【００５４】本発明の諸コンポーネントが分散されているとはいえ、本発明の動作は図１の形態と同じままである。例えば、RegisterImage()呼び出しは、コンピュータ４２上のアプリケーション１００から始動され、ネットワークをわたってコンピュータ４４に送られ画像サーバ１０で処理される。画像サーバ１０はハンドルを作成し、アプリケーション１００にハンドルを戻す。次に、

アプリケーション100は、コンピュータ44上のプリンタ・ドライバ300'による印刷ジョブを始動させるためネットワークを通して通信する。この時点から、同じコンピュータ44内で動作するプリンタ・ドライバ300'および画像サーバ10は、上述した形態と同様に相互に対話しながら最終的プリンタ画像を生成する。代替的形態としては、アプリケーション100は、コンピュータ48上のプリンタ・ドライバ300"を用いる印刷ジョブを始動することもできる。コンピュータ44上のプリンタ・ドライバ300'と同様に、コンピュータ48上のプリンタ・ドライバ300"は、画像サーバ10と画像処理要求について交渉する。しかしながら、コンピュータ44上のプリンタ・ドライバ300'と相違して、プリンタ・ドライバ300"はネットワークを通して画像サーバ呼び出しについて交渉しなければならない。

【0055】本発明には、例として次のような実施様態が含まれる。

（1）デジタル画像を取得するステップ(a)と、上記デジタル画像の所望の様相を標示する変換パラメータを指定するステップ(b)と、上記デジタル画像を画像サーバに登録するステップ(c)と、上記登録ステップ(c)に対する応答として上記画像サーバから画像識別子ハンドルを受け取るステップ(d)と、上記デジタル画像を識別する上記画像識別子と共に上記変換パラメータに従って上記デジタル画像の画像データの所望の様相を生成するように画像処理要求を上記画像サーバに送るステップ(e)と、生成された上記画像データを上記画像サーバから取り出すステップ(f)と、を含む画像処理方法。

（2）上記方法が、ユーザ・アプリケーションおよび上記画像サーバを並列かつ独立に実行するプログラムとして1つのコンピュータにロードするステップを更に含み、上記アプリケーションが上記（1）に記載のステップ(a)乃至(f)を実行する、上記（1）に記載の画像処理方法。

（3）上記方法が、第1および第2のユーザ・アプリケーションを1つのコンピュータにロードするステップを更に含み、上記第1のユーザ・アプリケーションが第1のデジタル画像に関して上記ステップ(a)乃至(e)を実行し、上記第2のユーザ・アプリケーションが第2のデジタル画像に関して上記ステップ(a)乃至(e)を実行する、上記（1）または（2）に記載の画像処理方法。

（4）上記方法が第1のコンピュータにユーザ・アプリケーションをロードし第2のコンピュータに上記画像サーバをロードするステップを更に含み、上記ユーザ・アプリケーションが、上記ステップ(a)ないし(f)を実行する、上記（1）、（2）または（3）に記載の画像処理方法。

【0056】（5）上記方法が、ユーザ・アプリケーション、上記画像サーバおよびプリンタ・ドライバをそれぞれ並列かつ独立に実行するプログラム単位としてロードするステップを更に含み、上記プリンタ・ドライバが第1のタイプのプリンタと通信し、上記ユーザ・アプリケーションが上記ステップ(a)乃至(d)を実行し、上記方法が更に上記ユーザ・アプリケーションから上記プリンタ・ドライバへ上記画像ハンドルおよび変換パラメータを送るステップを含み、その後上記プリンタが上記ステップ(d)を実行し、上記方法が上記プリンタ・ドライバから上記第1のタイプのプリンタへ上記画像データを送るステップを更に含む、上記（1）（2）、（3）または（4）に記載の画像処理方法。

（6）上記方法が、並列かつ独立に実行するプログラムとし上記コンピュータに第2のプリンタ・ドライバをロードするステップを更に含み、上記第2のプリンタ・ドライバが第2のタイプのプリンタと通信するように構成される、上記（5）に記載の画像処理方法。

（7）第1のコンピュータにユーザ・アプリケーションをロードし、上記画像サーバおよび上記プリンタ・ドライバを第2のコンピュータ上にロードするステップを更に含む、上記（5）に記載の画像処理方法。

【0057】（8）ドキュメント処理アプリケーションからドキュメントを印刷するシステムであって、ドキュメントの記憶位置を指定するドキュメント位置指定子を有する複数のドキュメントを含む第1の記憶装置と、それぞれ独立した実行プログラム単位である画像処理サーバおよびプリンタ・ドライバと、を備え、上記画像処理サーバが、上記ドキュメント位置指定子を受領次第上記ドキュメント識別子によって指定されるドキュメントを識別するドキュメント識別子を生成する電気回路を持ち、該システムは更に、上記ドキュメント位置指定子のうちの1つを上記画像処理サーバに通知するコード、上記画像処理サーバから対応するドキュメント識別子を取得するコード、および印刷パラメータおよび上記対応するドキュメント識別子を含むジョブ・パラメータを上記プリンタ・ドライバに通知するコードを含む印刷ジョブ始動コードを備え、上記プリンタ・ドライバが、上記印刷パラメータに従って上記画像処理サーバに画像処理要求を送るコードを含め、対応するドキュメント識別子によって識別されるドキュメントの画像データを操作するコード、上記画像処理サーバから上記操作された画像データを取得するコード、および上記操作された画像データを第1のタイプのプリンタに送るコードを有する、ドキュメント印刷システム。

（9）上記ドキュメント処理アプリケーション、画像処理サーバおよびプリンタ・ドライバが、単一のコンピュータに配置される、上記（8）に記載のドキュメント印刷システム。

【0058】

【発明の効果】本発明に従ったドキュメント識別子の使用によって、印刷の間システムを通過するデータ量は最

小限に保たれる。また、本発明によって、最終的プリンタ画像を形成するために必要とされる画像部分だけが画像サーバによってアクセスされるので、ドキュメントに含まれる画像の取り扱い量は最小限にとどめられる。この結果、プリンタ画像の形成の間にディスクとの間で行われるデータ保存および読み出しのためのディスク活動が削減され、ＣＰＵに対する負荷も減少し、従って、全般的システム性能が向上するという効果を奏する。

【図面の簡単な説明】

【図１】本発明の実施形態のシステムを示すブロック図である。

【図２】ドキュメント処理の際の本発明に従う典型的事象シーケンスを示すブロック図である。

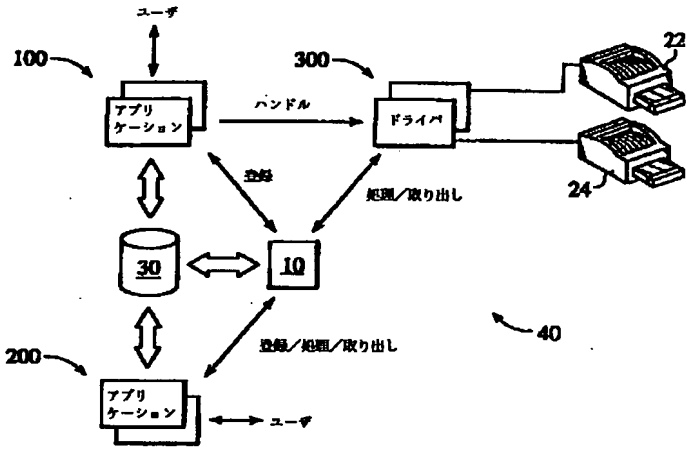
【図３】本発明のシステムのソフトウェア・コンポーネントの間のインターフェースを示すブロック図である。

【図４】ネットワーク構成に配置された本発明のシステムを示すブロック図である。

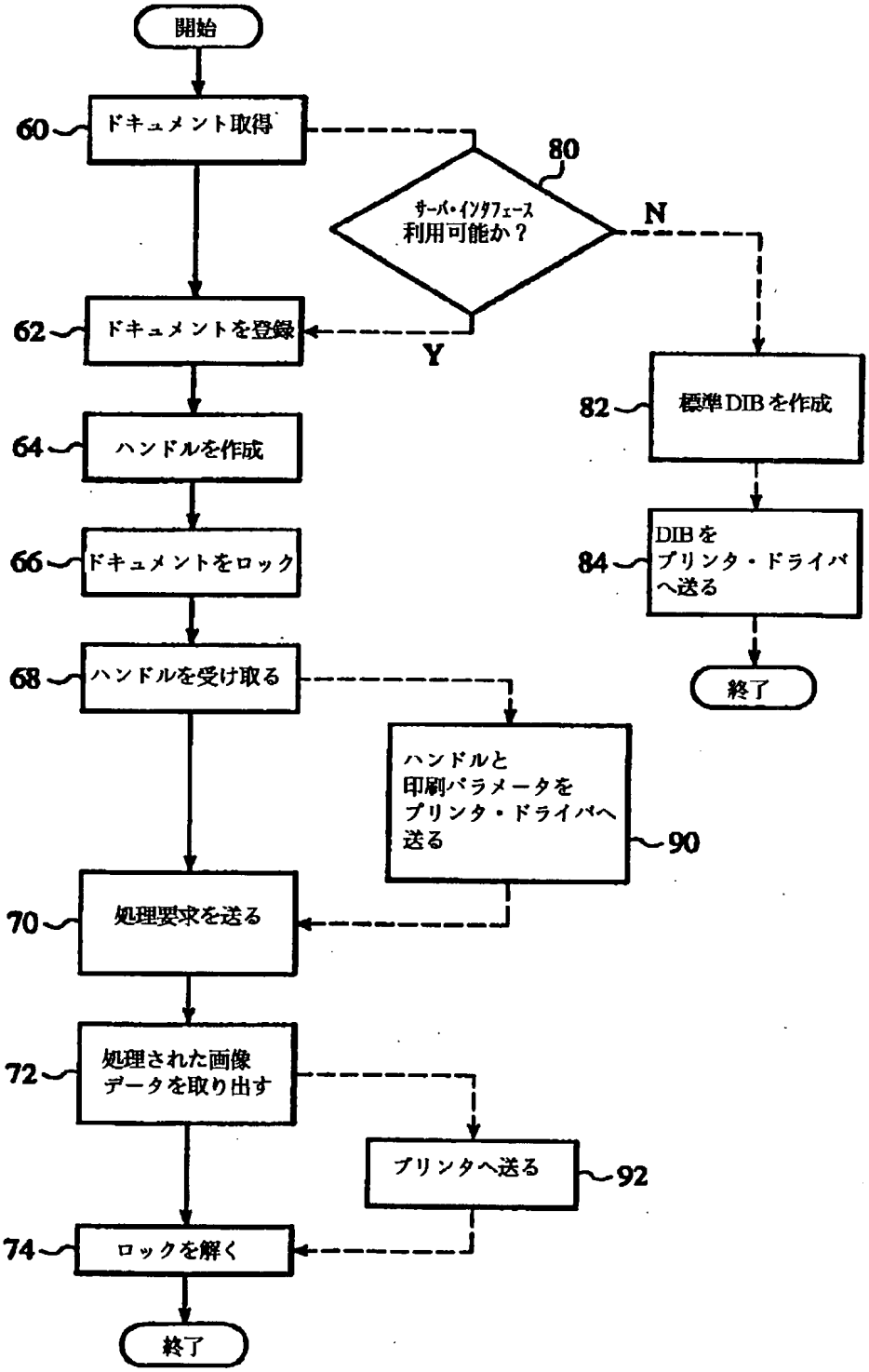
【符号の説明】

１０　画像サーバ
１１　画像ドライバ特有コード
１２　画像サーバ・ユーティリティ
２２、２４、２６、２８　プリンタ
３０　データ記憶装置
３２　Ｉ／Ｏユーティリティ
４０、４２、４４、４６、４８　コンピュータ・システム
５０　ネットワーク
５２　ネットワーク・インターフェース
１００、２００　アプリケーション
１０１　アプリケーション特有コード
３００、３００’、３００”　プリンタ・ドライバ
３０１　ドライバ特有コード
３０２　プリンタ・ユーティリティ

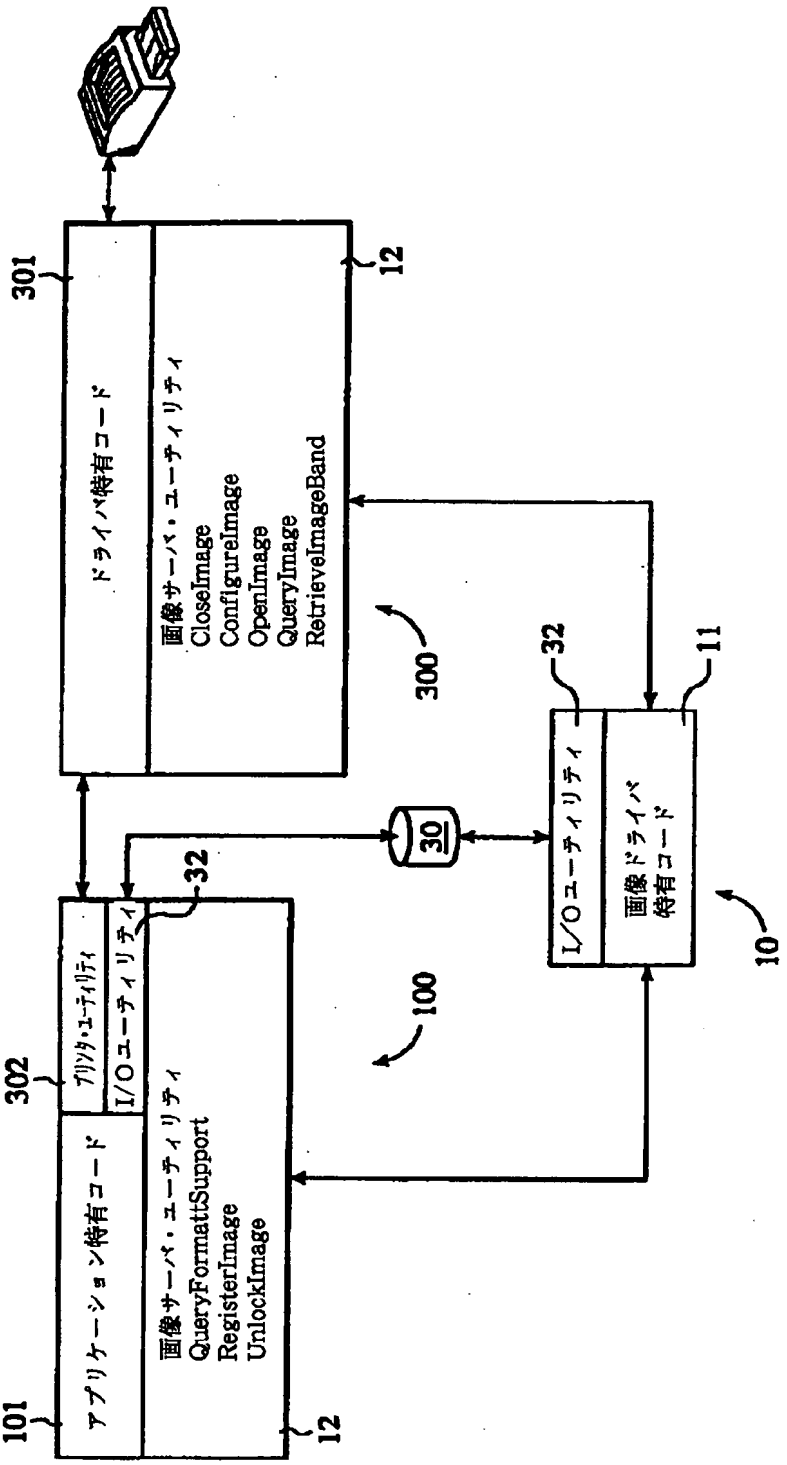
【図１】



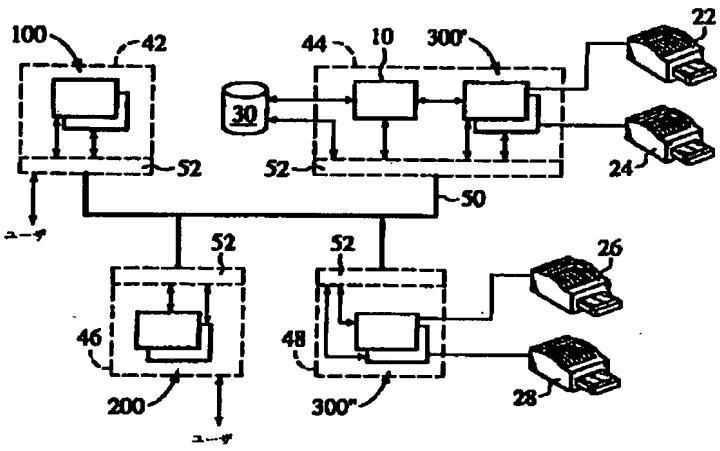
【図2】


開始
60
ドキュメント取得
80
サーバ・インタフェース
利用可能か？
N
Y
62
ドキュメントを登録
64
ハンドルを作成
66
ドキュメントをロック
68
ハンドルを受け取る
82
標準DIBを作成
84
DIBを
プリンタ・ドライバ
へ送る
終了
ハンドルと
印刷パラメータを
プリンタ・ドライバへ
送る
90
70
処理要求を送る
72
処理された画像
データを取り出す
プリンタへ送る
92
74
ロックを解く
終了

【図3】


301
ドライバ特有コード
画像サーバ・ユーティリティ
CloseImage
ConfigureImage
OpenImage
QueryImage
RetrieveImageBand
12
300
32
11
I/Oユーティリティ
画像ドライバ
特有コード
10
30
32
100
302
プリンタ・ユーティリティ
I/Oユーティリティ
アプリケーション特有コード
画像サーバ・ユーティリティ
QueryFormattSupport
RegisterImage
UnlockImage
101
12

【図4】



フロントページの続き

(72)発明者　ロバート・チョウ
アメリカ合衆国92128カリフォルニア州サン・ディエゴ、ゲーブル・リッヂ・ロード　14844

(72)発明者　サチン・ネイク
アメリカ合衆国92122カリフォルニア州サン・ディエゴ、カミント・ディア　7949、ナンバー　4